Joachim Rang

Apdative timestep control for fully implicit Runge–Kutta methods of higher order

Informatikbericht Nr. 2014-03

Institute of Scientific Computing
Carl-Friedrich-Gauss-Fakultät
Technische Universität Braunschweig

Braunschweig, Germany

This document was created February 2014 using LaTeX 2_ε.

Institute of Scientific Computing
Technische Universität Braunschweig
Hans-Sommer-Straße 65
D-38106 Braunschweig, Germany

url: www.wire.tu-bs.de
mail: wire@tu-bs.de

Apdative timestep control for fully implicit Runge–Kutta methods of higher order

Joachim Rang

Institute of Scientific Computing, TU Braunschweig
j.rang@tu-bs.de

Abstract

It is possible to construct fully implicit Runge–Kutta methods like Gauß-Legendre, Radau-IA, Radau-IIA, Lobatto-IIIA, -IIIB, and -IIIC methods of arbitrary high order of convergence. The aim of this paper is to find a new adaptive time stepping for these classes which is based on the embedding technique. Adaptive time step control with embedding is well-known for Runge–Kutta methods, and therefore new embedded methods of order $s-1$ for the above classes of fully implicit Runge–Kutta methods are constructed.

Since these fully implicit methods need the solution of a huge non-linear system of equations different approaches for non-linear equations are discussed and compared. It can be observed that non-linear solvers like the usually used simplified Newton method have a step size restriction if they are applied on higher order methods.

We apply our new methods on some lower dimensional ODEs to show that our approach leads to an efficient method.

1 Introduction

In the numerical solution of ordinary differential equations (ODEs) mainly two classes of numerical methods are known: one-step and multi-step methods, which may be explicit or implicit methods. One-step methods have the advantage that higher order methods can be created without having stability problems as known from multi-step methods. Moreover, they often allow an easy implementation of adaptive time step control. A disadvantage of one-step methods is the order reduction if they are applied on stiff ODEs like the example of Prothero–Robinson [27] or on differential algebraic equations (DAEs) [14].

In this paper we concentrate on fully implicit Runge–Kutta methods of higher order, which allow an adaptive time step control with the embedding technique. In every time step a non-linear system of dimension ns has to be solved, where n is the dimension of the problem and s is the number of internal stages of the Runge–Kutta method. In the last decades several papers discussed the efficient solution of the non-linear and linear equations. First [4] and [2] introduced a transformation of the coefficient matrix of the Runge–Kutta method splits. If a simplified Newton method is applied this splitting leads to s (may be complex valued) systems of dimension n. An application of this technique for Radau methods can be found in [15]. These non-linear systems can be solved directly with the help of LU-decompositions and back- and forward substitutions. But here we have to store s LU-decompositions which might be sometimes impossible. Therefore in [23] and [25] an iterative solution technique for the linear systems is applied. Another possibility offers the so-called *approximate Matrix factorisation* (AMF), which was introduced in [26]. An adaptive solution technique with AMF and a Newton-type iteration is suggested in [12]. Jay introduced in [16] and [17] *super partitioned additive Runge–Kutta methods* (SPARK methods). In [13] fully implicit Runge–Kutta methods are solved with a single Newton method.

An adaptive time step control may undoubtedly improve the accuracy and efficiency of simulations substantially. One possibility to achieve adaptivity in time is Richardson's extrapolation [14, 30]. In this case every time step is computed twice, first with step size τ and second with step size $\tau/2$. With the help of the two approximations a numerical error can be found which implies the step size for the next time step. This approach can be applied on every one-step method, but in each time step the number of non-linear systems increases by a factor of three. A much cheaper possibility is the so-called embedding technique where the second solution is computed with almost the same coefficients, such that no further costs for linear algebra arise [14, 30]. In practice this technique is only applied if the order of both methods differ

by only one. Therefore Hairer and Wanner suggest in [15] a new step size control for Raudau-methods where an automatic choice of Radau methods of different order is integrated. An improvement for a Radau-IIA method of order 17 can be found in [24]. In this paper adaptivity is achieved by using the embedding technique, i. e. an embedded method of order $s-1$ is created and then we apply the PI-controller as usual (see [14, 21]).

In this paper we start our considerations with Gauß-Legendre, Radau-IA, Radau-IIA, Lobatto-IIIA, -IIIB and -IIIC methods. These methods can be created with arbitrary large convergence order (see [14, 30]). For all of these methods we develop embedded methods of order $s-1$. In Chapter 3 we consider the simplified Newton method for solving the arising non-linear systems. Since this system is of dimension ns, where n is the dimension of the ODE and s is the number of internal stages, we use the transformation of the coefficient matrix of the Runge–Kutta method into a complex valued diagonal matrix (see [4] and [2]) because we need only the solution of s non-linear systems of dimension n. From the book of Deuflhard [9] it is known that a step size restriction appears if the non-linear systems are solved with Newton methods. For lower order methods the maximal possible step size is dominated from the accuracy of the method and not from the non-linear solver. But if we increase the order of the method, it is possible to increase the time step size without loosing accuracy. This has the effect that for higher order method this step size restriction is rather important. In this paper we consider as a second solution technique a modified fixed-point iteration [19, 18]. This approach has the advantage that larger time steps are possible, but the benefit of a constant iteration matrix is lost. Finally we present some numerical results to show that the new adaptive time step control leads to an efficient solution strategy. We compare our numerical methods with some fourth order DIRK- and ROW methods to show that the suggested methods are a good alternative.

2 Implicit Runge–Kutta methods

We start our considerations with the initial value problem

$$\dot{\mathbf{u}} = \mathbf{f}(t, \mathbf{u}), \quad \mathbf{u}(t_0) = \mathbf{u}_0. \tag{1}$$

A Runge-Kutta method for the ODE (1) is given by

$$\mathbf{k}_i = \mathbf{f}\left(t_m + c_i\tau, \mathbf{u}_m + \tau_m \sum_{j=1}^{s} a_{ij}\mathbf{k}_j\right), \quad i = 1, \ldots, s, \tag{2}$$

$$\mathbf{u}_{m+1} = \mathbf{u}_m + \tau_m \sum_{i=1}^{s} b_i \mathbf{k}_i, \tag{3}$$

where τ is a given time step size, s is the number of internal stages and a_{ij}, b_i, and c_i are the coefficients of the RK-method, which should be determined in such a way that the method has a sufficiently high order convergence [5, 14, 30]. The order of the RK-method can be determined with the so-called simplifying conditions from Butcher [3], which are defined as follows.

Definition 2.1. *(see [3]). An s-stage RK-method satisfies the simplifying conditions, if the conditions*

$$\begin{aligned}
B(p): &\quad \sum_{i=1}^{s} b_i c_i^{k-1} &&= 1/k, && k = 1, \ldots, p,\\
C(q): &\quad \sum_{j=1}^{s} a_{ij} c_j^{k-1} &&= c_i^k/k, && i = 1, \ldots, s, k = 1, \ldots, q,\\
D(r): &\quad \sum_{i=1}^{s} b_i c_i^{k-1} a_{ij} &&= b_j(1 - c_j^k)/k, && j = 1, \ldots, s, k = 1, \ldots, r
\end{aligned}$$

are fulfilled.

The condition $B(p)$ is equivalent to a quadrature rule with nodes c_i and weights b_i, which integrates polynomials of degree $p-1$ exactly. The conditions $C(q)$ have the following meaning. The intermediate values $\mathbf{k}_i$ are integrated exactly by a quadrature rule with weights a_{ij} and nodes c_i, which integrates polynomials of degree q exactly.

Theorem 2.2. *(see [5, 30]) An RK-method with s internal stages has the convergence order p, if the simplifying conditions $B(p)$, $C(l)$, and $D(m)$ with*

$$p \leq \min\{l + m + 1, 2l + 2\}$$

are satisfied.

For the proof we refer to the book of Butcher [5].

2.1 Gauß–Legendre methods

In the following we derive the Gauß–Legendre methods (see [5, 14, 30]). In the case of the Gauß quadrature rules the nodes c_i are chosen as the roots of the shifted Legendre polynomial of degree s, i.e.

$$P_s(2t-1) = \frac{1}{s!}\frac{d^s}{dt^s}[t^s(t-1)^s].$$

With respect to the $L_2(0,1)$-scalar product the polynomial $P_s(2t-1)$ is orthogonal to all polynomials of degree $< s$. The roots of the Legrendre polynomials P_s can be found in the book of Abramowitz and Stegun [1] or can be computed with a computer algebra tool. It can be proven that the roots are pairwise distinct. From this fact it follows that the Vandermonde matrix

$$V_s = (V_{ij}) := (c_i^{j-1}) = \begin{pmatrix} 1 & c_1 & c_1^2 & \dots & c_1^{s-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & c_s & c_s^2 & \dots & c_s^{s-1} \end{pmatrix}, \qquad i,j = 1,\dots,s$$

is regular. The simplifying conditions $B(p)$ and $C(q)$ can be written in matrix-vector notation. The condition $B(p)$ reads as $\mathbf{b}^\top \mathbf{c}^k = 1/k$, where the vector $\mathbf{c}^k$ is defined as $\mathbf{c}^k = (c_1^k, \dots, c_s^k)^\top$. The condition $C(q)$ can be written as $A\mathbf{c}^{k-1} = \mathbf{c}^k/k$, where $A = (a_{ij})_{i,j=1}^s$. The nodes b_i are then uniquely determined by the conditions $B(1)$, ..., $B(s)$, i. e. by

$$\mathbf{b}^\top \mathbf{e} = 1, \mathbf{b}^\top \mathbf{c} = 1/2, \dots, \mathbf{b}^\top \mathbf{c}^{s-1} = 1/s.$$

This system can be written in matrix-vector notation as

$$\mathbf{b}^\top V_s = \mathbf{e}_H^\top := \left(1, \frac{1}{2}, \dots, \frac{1}{s}\right).$$

Multiplying from the right with the inverse of V_s generates us our nodes b_i, i.e. $\mathbf{b}^\top = \mathbf{e}_H^\top V_s^{-1}$. For the embedded method we set

$$\tilde{\mathbf{e}}_H^\top := \left(1, \frac{1}{2}, \dots, \frac{1}{s-1}, 0\right).$$

Then the nodes $\tilde{b}_i$ are given simply by

$$\tilde{\mathbf{b}}^\top = \tilde{\mathbf{e}}_H^\top V_s^{-1}$$

and the embedded method is of order $s-1$. Next we determine the matrix A with help of the conditions $C(l)$, $l = 1, \dots, s$. These conditions can be written as

$$A\mathbf{e} = \mathbf{c}, A\mathbf{c} = \mathbf{c^2}/2, \dots, A\mathbf{c}^{s-1} = \mathbf{c^s}/s,$$

or in matrix-vector notation by

$$AV_s = C, \quad C := (c_{ij}) = \frac{1}{j} c_i^j.$$

It follows $A = CV_s^{-1}$ and our Butcher table looks as follows

$$\begin{array}{c|c} \mathbf{c} & CV_s^{-1} \\ \hline & \mathbf{e}_H^\top V_s^{-1} \\ \hline & \tilde{\mathbf{e}}_H^\top V_s^{-1} \end{array}.$$

For $s = 2$ and $s = 3$ we get the following methods

$$\begin{array}{c|cc} \frac{1}{2} - \frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4} - \frac{\sqrt{3}}{6} \\ \frac{1}{2} + \frac{\sqrt{3}}{6} & \frac{1}{4} + \frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} + \frac{\sqrt{3}}{2} & \frac{1}{2} - \frac{\sqrt{3}}{2} \end{array} \qquad \begin{array}{c|ccc} \frac{1}{2} - \frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9} - \frac{\sqrt{15}}{10} & \frac{5}{36} - \frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36} + \frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36} - \frac{\sqrt{15}}{24} \\ \frac{1}{2} + \frac{\sqrt{15}}{10} & \frac{5}{36} - \frac{\sqrt{15}}{30} & \frac{2}{9} + \frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \\ \hline & -\frac{5}{6} & \frac{8}{3} & -\frac{5}{6} \end{array}$$

Theorem 2.3. *The Gauß-Legendre method with s stages has order $p = 2s$.*

For the proof we refer to [5, 14, 30].

2.2 Radau-IA- and IIA-methods

It can be shown that methods of order $2s$ have not the best stability properties. Therefore we look now for methods of order $2s - 1$ and start with the following theorem:

Theorem 2.4. *Let be given a Runge–Kutta method with $p = 2s - 1$. Then the c_i's are given by the roots of the polynomial*

$$P_{s,\xi}(2x - 1) = P_s(2x - 1) + \xi P_{s-1}(2x - 1), \quad \xi \in \mathbb{R}.$$

Proof. see [31]. □

Here we are interested in the cases $\xi = +1$ (Radau-I methods with $c_1 = 0$) and $\xi = -1$ (Radau-II methods with $c_s = 1$).

The weights b_i are determined for both classes with the simplifying conditions $B(1), \ldots, B(s)$. In the case of the *Radau-IIA-methods* (see [10]) the matrix A is determined with $C(s)$. For the *Radau-IA-methods* (see [10]) the

coefficient matrix A are determined with the condition $D(s)$, which can be written as s linear equations of dimension s. The j-th column of A can be determined by solving the equations

$$\begin{pmatrix} b_1 & b_2 & \dots & b_s \\ b_1c_1 & b_2c_2 & \dots & b_sc_s \\ \vdots & \vdots & & \vdots \\ b_1c_1^{s-1} & b_2c_2^{s-1} & \dots & b_sc_s^{s-1} \end{pmatrix} \begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{sj} \end{pmatrix} = \begin{pmatrix} 1-c_j \\ (1-c_j)/2 \\ \vdots \\ (1-c_j)/s \end{pmatrix}.$$

In both cases the nodes $\tilde{b}_i$ of the embedded method are determined by $\tilde{\mathbf{b}}^\top = \tilde{\mathbf{e}}_H^\top V_s^{-1}$. This setting leads to embedded methods of order $s-1$. For $s=2$ and $s=3$ we get the following Radau-IA methods

$$\begin{array}{c|cc} 0 & \frac{1}{4} & -\frac{1}{4} \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} \\ \hline & \frac{1}{4} & \frac{3}{4} \\ \hline & 1 & 0 \end{array} \qquad \begin{array}{c|ccc} 0 & \frac{1}{9} & -\frac{1+\sqrt{6}}{18} & \frac{-1+\sqrt{6}}{18} \\ \frac{6-\sqrt{6}}{10} & \frac{1}{9} & \frac{88+7\sqrt{6}}{360} & \frac{88-43\sqrt{6}}{360} \\ \frac{6+\sqrt{6}}{10} & \frac{1}{9} & \frac{88+43\sqrt{6}}{360} & \frac{88-7\sqrt{6}}{360} \\ \hline & \frac{1}{9} & \frac{16+\sqrt{6}}{36} & \frac{16-\sqrt{6}}{36} \\ \hline & -1 & \frac{1}{72}(6+\sqrt{6})^2\sqrt{6} & -\frac{1}{72}(-6+\sqrt{6})^2\sqrt{6} \end{array}.$$

The Radau-IIA methods with 2 and 3 internal stages are given by

$$\begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \\ \hline & 1 & 0 \end{array} \qquad \begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2+3\sqrt{6}}{225} \\ 1 & \frac{16-6\sqrt{6}}{36} & \frac{16+6\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{16-6\sqrt{6}}{36} & \frac{16+6\sqrt{6}}{36} & \frac{1}{9} \\ \hline & -1 & 1-\frac{7}{12}\sqrt{6} & 1+\frac{7}{12}\sqrt{6} \end{array}.$$

2.3 Lobatto-IIIA-, -III-B-, and -IIIC-methods

Next we are interested in the construction of methods which have order $2s-2$. Therefore we need the following theorem.

Theorem 2.5. *Let be given a Runge–Kutta method with $p = 2s-2$. Then the nodes c_i are given by the roots of the polynomial*

$$P_{s,\xi,\mu}(2x-1) = P_s(2x-1) + \xi P_{s-1}(2x-1) + \mu P_{s-2}(2x-1), \quad \xi, \mu \in \mathbb{R}.$$

Proof. see [31]. □

If we consider the cases $\xi = 0$ and $\mu = 1$ we obtain the so-called Lobatto-III methods which have the properties $c_1 = 0$ and $c_s = 1$. The weights b_i are determined with the simplifying conditions $B(1), \ldots, B(s)$ and the coefficient matrix A as follows:

- *Lobatto-IIIA* (see [10]): with $C(s)$
- *Lobatto-IIIB* (see [10]): with $D(s)$
- *Lobatto-IIIC* (see [6]): with $C(s-1)$ and $a_{i1} = b_1$, $i = 1, \ldots, s$.

The construction of the Lobatto-IIIA- and -IIIB-methods can be done in a similar way as in the last section. In the case of Lobatto-IIIC-methods the coefficient matrix A can be found in the following way. We consider the condition $C(i)$, i. e. $A\mathbf{c}^{k-1} = \mathbf{c}^k/k$. Since $c_1 = 0$ by construction we get

$$\sum_{j=2}^{s} a_{ij} c_j^{k-1} = \frac{c^k}{k}, \quad k = 2, \ldots, s$$

and finally the linear system

$$\begin{pmatrix} 1 & 1 & \ldots & 1 \\ c_2 & c_3 & \ldots & c_s \\ \vdots & \vdots & & \vdots \\ c_2^{s-2} & c_3^{s-2} & \ldots & c_s^{s-2} \end{pmatrix} \begin{pmatrix} a_{i2} \\ a_{i3} \\ \vdots \\ a_{is} \end{pmatrix} = \begin{pmatrix} 0 \\ c_2/2 \\ \vdots \\ c_{s-1})/(s-1) \end{pmatrix}.$$

For $s = 2$ and $s = 3$ we get the following Lobatto-IIIA methods

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \\ & 1 & 0 \end{array} \qquad \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{24} & \frac{1}{3} & -\frac{1}{24} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & -\frac{1}{2} & 2 & -\frac{1}{2} \end{array}.$$

The Lobatto-IIIB methods with 2 and 3 internal stages are given by

$$\begin{array}{c|cc} 0 & \frac{1}{2} & 0 \\ 1 & \frac{1}{2} & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \\ \hline & 1 & 0 \end{array} \qquad \begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{6} & 0 \\ \frac{1}{2} & \frac{1}{6} & \frac{1}{3} & 0 \\ 1 & \frac{1}{6} & \frac{5}{6} & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & -\frac{1}{2} & 2 & -\frac{1}{2} \end{array}.$$

For $s = 2$ and $s = 3$ we get the following Lobatto-IIIC methods

$$\begin{array}{c|cc} 0 & \frac{1}{2} & -\frac{1}{2} \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \\ \hline & 1 & 0 \end{array} \qquad \begin{array}{c|ccc} 0 & \frac{1}{6} & -\frac{1}{3} & \frac{1}{6} \\ \frac{1}{2} & \frac{1}{6} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \hline & -\frac{1}{2} & 2 & -\frac{1}{2} \end{array}.$$

3 Solution of the non-linear systems

If we apply an implicit Runge–Kutta method to a system of ODEs a large non-linear system of equations has to be solved in every time step. We start our considerations with the implicit Runge–Kutta method (2)–(3), which can be written as

$$\begin{pmatrix} \mathbf{k}_1 \\ \vdots \\ \mathbf{k}_s \end{pmatrix} = \begin{pmatrix} \mathbf{f}\left(t_m + c_1\tau, \mathbf{u}_m + \tau \sum_{j=1}^{s} a_{1j}\mathbf{k}_j\right) \\ \vdots \\ \mathbf{f}\left(t_m + c_s\tau, \mathbf{u}_m + \tau \sum_{j=1}^{s} a_{sj}\mathbf{k}_j\right) \end{pmatrix}. \tag{4}$$

Equation (4) forms a non-linear system of equations which has dimension $ns \times ns$. There are different possibilities to solve this system. A fixed-point iteration might not be a good idea, since for stiff problems a step size restriction can be observed (see [14, 31]). Therefore often a simplified Newton method for computing the numerical solution of (4) is used, which is considered in Section 3.1. Since this system is of dimension ns a transformation of the coefficient matrix A can be applied, which splits this large system into s smaller ones (see [2] and [4]). In our case these linear systems are complex. A convergence theorem from Deuflhard [9] states that the simplified

Newton method converges if the step size is sufficiently small. Moreover, an upper bound for the step size is given in [9], which is independent of the Runge–Kutta method and of the order of the method.

The Newton method method has the disadvantage that often numerical instabilities arise, i.e. with a bad starting value convergence may not be archived or as, in our case, one gets an upper bound for the step size, which might be rather small in comparison to the step size which might be possible since the method has a high order of convergence. In extreme cases it is possible that either the Newton method diverges or the accuracy of the numerical approximation is in the range of the machine accuracy. Therefore in practical applications sometimes a modified fixed-point iteration is applied (see [18, 19]), which we consider in Section 3.4. But in every iteration step the modified fixed-point iteration needs the solution of the huge non-linear system of equations. Moreover, the Jacobian of the system changes and therefore an LU decomposition with forward and backward substitutions can not be applied. In the comparison in Section 3.5 we see that the bound for the step size in the case of modified fixed-point iteration is much larger than in the case of Newton's method, but the computing time for the modified fixed-point iteration is much longer.

3.1 A simplified Newton method

In this section we apply the simplified Newton method on the non-linear system (4). By $\mathbf{f}_u$ we denote the Jacobian of $\mathbf{f}$ at the point $(t_m, \mathbf{u}_m)$ and then the simplified Newton method reads as

$$\begin{pmatrix} I - \tau a_{11}\mathbf{f}_u & \dots & -\tau a_{1s}\mathbf{f}_u \\ \vdots & & \vdots \\ -\tau a_{s1}\mathbf{f}_u & \dots & I - \tau a_{ss}\mathbf{f}_u \end{pmatrix} \begin{pmatrix} \Delta\mathbf{k}_1^{(\nu+1)} \\ \vdots \\ \Delta\mathbf{k}_s^{(\nu+1)} \end{pmatrix} = \begin{pmatrix} \mathbf{f}\left(t_m + c_1\tau, \mathbf{u}_m + \tau\sum_{j=1}^{s} a_{1j}\mathbf{k}_j^{(\nu)}\right) \\ \vdots \\ \mathbf{f}\left(t_m + c_s\tau, \mathbf{u}_m + \tau\sum_{j=1}^{s} a_{sj}\mathbf{k}_j^{(\nu)}\right) \end{pmatrix} - \begin{pmatrix} \mathbf{k}_1^{(\nu)} \\ \vdots \\ \mathbf{k}_s^{(\nu)} \end{pmatrix} \tag{5}$$

with $\Delta\mathbf{k}_i^{(\nu+1)} = \mathbf{k}_i^{(\nu+1)} - \mathbf{k}_i^{(\nu)}$, $i = 1, \ldots, s$. If we introduce the Kronecker symbol $A \otimes B$ (see [30]) defined by

$$A \otimes B = \begin{pmatrix} a_{11}B & \ldots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \ldots & a_{mn}B \end{pmatrix},$$

where A and B are matrices, we can write our linear systems as

$$(I_s \otimes I_n - \tau A \otimes \mathbf{f}_u)\Delta\mathbf{K}^{(\nu+1)} = \mathbf{F}^{(\nu)} \tag{6}$$

with $\Delta\mathbf{K}^{(\nu+1)} := (\Delta\mathbf{k}_1^{(\nu+1)}, \ldots, \Delta\mathbf{k}_s^{(\nu+1)})^\top$ and

$$\mathbf{F}^{(\nu)} := \begin{pmatrix} \mathbf{f}\left(t_m + c_1\tau, \mathbf{u}_m + \tau\sum_{j=1}^s a_{1j}\mathbf{k}_j^{(\nu)}\right) \\ \vdots \\ \mathbf{f}\left(t_m + c_s\tau, \mathbf{u}_m + \tau\sum_{j=1}^s a_{sj}\mathbf{k}_j^{(\nu)}\right) \end{pmatrix} - \begin{pmatrix} \mathbf{k}_1^{(\nu)} \\ \vdots \\ \mathbf{k}_s^{(\nu)} \end{pmatrix}.$$

The linear system (6) can be solved with only one LU decomposition and then in each Newton iteration a forward and a backward substitution can be applied, since the coefficient matrix does not change during the time step.

As starting values for the Newton method often the setting $\mathbf{k}_i^{(0)} := 0$ is used. This is of course not the best choice. In [14] the starting values are computed with the help of interpolation. Therefore let

$$\mathbf{z}_i := \mathbf{U}_i - \mathbf{u}_m = \tau\sum_{j=1}^s a_{ij}\mathbf{k}_j.$$

Then it holds

$$\mathbf{z}_i = \mathbf{u}(t_m + c_i\tau_m) - \mathbf{u}_m + \mathcal{O}(\tau^{\eta+1}),$$

if the simplifying condition $C(\eta)$ is satisfied for some $\eta \leq s$. In the case of the Gauß-Legendre and the Radau-IIA methods we have $c_1 \neq 0$ and we can consider the interpolation polynomial of degree s, defined by

$$\mathbf{q}(0) = 0, \mathbf{q}(c_i) = \mathbf{z}_i, \quad i = 1, \ldots, s,$$

since $c_i = \sum_{j=1}^s a_{ij}$ holds (see simplifying condition $C(1)$). In the other cases an interpolation polynomial of degree $s-1$ can be applied. In our case we start with a zero starting value since a polynomial interpolation often leads to oscillations.

3.2 A transformation of the coefficient matrix

Since the solutions of the huge non-linear systems are very expensive, we try to reduce these costs by splitting these systems into s smaller ones. One possibility is a transformation of coefficient matrix A, which was derived independently from Bickart [2] and Butcher [4]. We multiply system (6) from the left with $P \otimes I$ and from the right with $Q \otimes I$. Matrices P and Q are chosen in such a way that the product

$$(P \otimes I)(I_s \otimes I_n - \tau A \otimes \mathbf{f}_u)(Q \otimes I)$$

is a lower block triangular matrix. Let us assume that the coefficient matrix A is regular, which is true for the Gauß-Legendre, the Radau-IA, the Radau-IIA and the Lobatto-IIIC methods. Then the eigenvalues of the coefficient matrix A are non-zero and it is possible to compute the Jordan canonical of A, which is given by

$$T^{-1}A^{-1}T = \begin{pmatrix} \lambda_1^{-1} & & 0 & \\ \mu_1 & \lambda_2^{-1} & & \\ & \ddots & \ddots & \\ 0 & & \mu_{s-1} & \lambda_s^{-1} \end{pmatrix},$$

where $\mu_i = 0$, if $\lambda_i \neq \lambda_{i+1}$. Next we introduce the diagonal matrix D given by $D := \operatorname{diag}(\lambda_1, \ldots, \lambda_s)$ and select $P := DT^{-1}A^{-1}$, $Q := T$. Then we have

$$PQ = \begin{pmatrix} 1 & & 0 & \\ \epsilon_1 & 1 & & \\ & \ddots & \ddots & \\ 0 & & \epsilon_{s-1} & 1 \end{pmatrix}, \qquad \epsilon_i \in \{0, 1\}.$$

Note that the matrices D, P and Q are complex valued matrices in our cases.

3.3 Convergence of the simplified Newton method

Next we ask for which time step sizes τ do the simplified Newton method converges. From [22] we know that for sufficiently small τ convergence is achieved. But what happens for larger τ, especially if we use a higher order method? The answer is given by the following theorem, which can be found in the book of Deuflhard [9]:

Theorem 3.1. *Let $D \subset \mathbb{R}^n$ be an open set and $\mathbf{f} : D \to \mathbb{R}^n$ be a continuously differentiable function. For the Jacobian $\mathbf{f}_u(t, \mathbf{u})$ we assume that a one-sided Lipschitz condition is satisfied with the one-sided Lipschitz constant μ. Let $\|\cdot\|$ be some norm and assume that*

$$\begin{aligned} \|\mathbf{f}(t_m, \mathbf{u}_m)\| &\leq L_0, \quad \mathbf{u}_m \in D, \\ \|(\mathbf{f}_u(t, \mathbf{u}) - \mathbf{f}_u(t_m, \mathbf{u}(t_m))\mathbf{v}\| &\leq L_2 \|\mathbf{u} - \mathbf{u}_m\| \|\mathbf{v}\|, \quad \mathbf{u}, \mathbf{u}_m, \mathbf{v} \in D. \end{aligned}$$

If D is sufficiently large then existence and uniqueness of the solution of the ODE is guaranteed in $[0, \bar{t}]$ such that

$$\begin{aligned} \tau \text{ unbounded, if } \mu\bar{\tau} &\leq -1, \\ \tau \leq \bar{\tau}\Psi(\mu\bar{\tau}), \text{ if } \mu\bar{\tau} &> -1, \end{aligned}$$

where $\bar{\tau} := 1/\sqrt{2L_0L_2}$ and

$$\Psi(s) := \begin{cases} \ln(1+s)/s, & s \neq 0, \\ 1, & s = 0 \end{cases}.$$

Proof. For the proof we refer to [9]. □

Example 3.1. *Let us consider the ODE*

$$\dot{u} = -5tu^2 + 5/t - 1/t^2, \quad u(t_0) = 1/t_0, t_0 > 0, \tag{7}$$

which has the exact solution $u(t) = 1/t$. The derivatives of the right-hand side of (7) w.r.t. u are given by $f_u = -10tu$ and $f_{uu} = -10t$. In this case we have $\mu = -10t$, $L_2 = 10$, and $L_0 = 1/t_0^2$. It follows

$$\bar{\tau} = (2L_0L_2)^{-1/2} = -1/\sqrt{20/t_0^2} = t_0/\sqrt{20}.$$

Since $\mu = -10t_0$ we have $\mu\bar{\tau} = -1/(t_0\sqrt{20})$ and we have step size restriction with

$$\tau \leq \frac{\ln(1+\mu\bar{\tau})}{\mu} = -\ln\left(1 - 10t_0^2/\sqrt{20}\right) \approx 0.02614.$$

In practice larger step-sizes can be used as those, which are estimated from Theorem 3.1.

3.4 A modified fixed-point iteration

As we have seen in the last subsection the simplified Newton method needs rather small time steps. Although in practise larger step-sizes may be used, we consider here another approach, which is called modified fixed-point iteration. Therefore we have to rewrite the general non-linear ODE (1) in the form

$$\dot{\mathbf{u}} = A(\mathbf{u})\mathbf{u} + \mathbf{f}(t), \quad \mathbf{u}(t_0) = \mathbf{u}_0. \tag{8}$$

Then we apply our implicit Runge–Kutta method (2)–(3) on the ODE (8) and get the non-linear system

$$\mathbf{k}_i = \mathbf{f}(t_m + c_i\tau) + A\left(\mathbf{u}_m + \tau_m \sum_{j=1}^{s} a_{ij}\mathbf{k}_j\right)\left(\mathbf{u}_m + \tau_m \sum_{j=1}^{s} a_{ij}\mathbf{k}_j\right), \tag{9}$$

for $i = 1, \ldots, s$. This system is now solved with a modified fixed-point iteration. The basic idea of this method is to use the old iterated value $\mathbf{k}_i^{(\nu)}$ in the argument of A in the right-hand side and the new iterated value $\mathbf{k}_i^{(\nu+1)}$ for the other term. Then we get an iterative method of the form

$$\begin{pmatrix} I - \tau_m a_{11} A\left(\mathbf{U}_1^{(\nu)}\right) & \ldots & -\tau_m a_{1s} A\left(\mathbf{U}_1^{(\nu)}\right) \\ \vdots & \ddots & \vdots \\ -\tau_m a_{s1} A\left(\mathbf{U}_s^{(\nu)}\right) & \ldots & I - \tau_m a_{ss} A\left(\mathbf{U}_s^{(\nu)}\right) \end{pmatrix} \begin{pmatrix} \mathbf{k}_1^{(\nu+1)} \\ \vdots \\ \mathbf{k}_s^{(\nu+1)} \end{pmatrix}$$
$$= \begin{pmatrix} \mathbf{f}_1(t_m + c_1\tau_m) - A\mathbf{u}_m \\ \mathbf{f}_2(t_m + c_2\tau_m) - A\mathbf{u}_m \\ \vdots \\ \mathbf{f}_s(t_m + c_s\tau_m) - A\mathbf{u}_m \end{pmatrix}, \tag{10}$$

where

$$\mathbf{U}_i^{(\nu)} := \mathbf{u}_m + \tau_m \sum_{j=1}^{s} a_{ij}\mathbf{k}_j^{(\nu)}, \quad i = 1, \ldots, s.$$

This iteration is called *modified fixed-point iteration*. This version of the fixed-point iteration is, for example, applied in the numerical solution of the incompressible Navier–Stokes equations [18]. Convergence can be proved by Banach's fixed-point theorem. As in the case of the simplified Newton method a step size restriction occurs, but in our numerical experiences this step size

restriction is much larger than the step size restriction of the simplified Newton method. The disadvantage of this approach is that the iteration matrix changes during the iteration, i.e. the costs for the linear algebra are much higher for the modified fixed point iteration than for the simplified Newton method.

3.5 Comparison of the approaches

Let us now compare the simplified Newton method with the modified fixed-point iteration. We consider the ODE (7) from Example 3.1 and want to find the maximal step size to get a stable numerical approximation for the first time step. Stable in this case means, that the numerical error should be smaller than 1/10. Therefore we compute the numerical solution after one time step and use 40 iterations. In Figure 1 the numerical results are presented and the methods applied with Newton's method are marked with "(N)". If the modified fixed-point iteration is used, the method is marked with "(F)". It can be observed that both approaches have a step size restriction. For the fixed-point iteration this step size restriction (≈ 4.0) is much larger than for Newton's method (≈ 1.0). This step size restriction is independent of the method, as Figure 1 shows. Moreover for small s the step size restriction is dominated by the numerical error. If we used less iterations for the nonlinear solvers the maximal time steps would become smaller. For example: with 5 iterations we get an upper bound of 0.43 for Newton's method and 0.41 for the modified fixed-point iteration. If we compare the efficiency of both approaches, we first mention the main advantage of the modified fixed-point iteration, which are higher possible step-sizes. But the convergence velocity of the modified fixed-point iteration is much slower than the convergence speed of the simplified Newton method. Moreover in the case of the modified fixed-point iteration in each iteration step we have to assemble the large iteration matrix of dimension ns. A transformation to s systems of dimension n is not possible.

4 Numerical examples for ODEs

In this section we apply the implicit Runge–Kutta methods with their new embedded formulas to two test problems. The first one is a chemical reaction problem (see [11]) and the the second one is the perturbed Kepler's problem. We compare these high order methods with a 4th order DIRK method, which was developed by Kvaerno [20], and with the RODASPR method from Rang [28] to show that higher order methods may be more efficient than lower

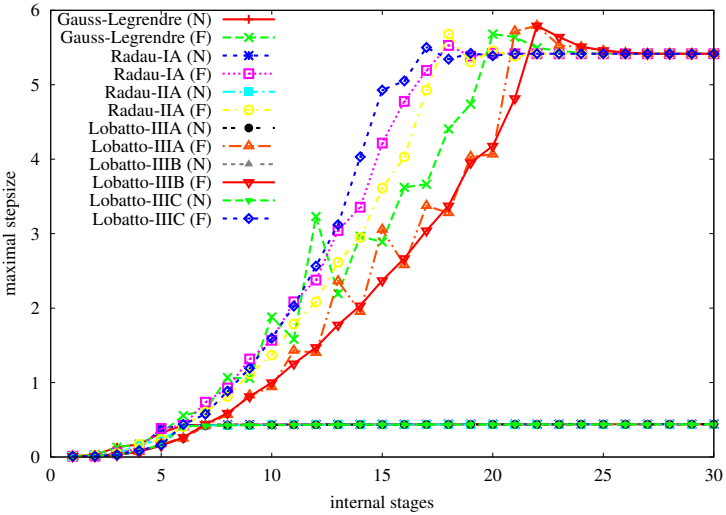


Figure 1: Comparison of Newton's method (N) with fixed-point iteration (F)

order methods although the costs of the linear algebra are very high. Since we consider only first order ODEs of dimension 4 the non-linear systems are solved directly with a simplified Newton method and an LU-decomposition. For this decomposition the use the UMFPACK software (see [8, 7]).

4.1 The perturbed Kepler problem

Consider the second order ODE

$$\ddot{y}_i = -\frac{y_i}{(y_1^2 + y_2^2)^{3/2}} - \epsilon \frac{3y_i}{2(y_1^2 + y_2^2)^{5/2}}, \quad i = 1, 2,$$

where ϵ is a small perturbation (see [29]). If $\epsilon = 0$ we get the original Kepler problem. The initial conditions are given by

$$y_1 = 0, \quad y_2 = \sqrt{\frac{1+e}{1-e}}, \quad \dot{y}_1 = 1 - e, \quad \dot{y}_2 = 0,$$

where $e \in [0, 1)$ is a parameter (see [29]).

For the unperturbed Kepler problem the parameter e represents the eccentricity of the ellipse. The period of the solution is equal to 2π and we have

$r_{max} = 1 + e$ and $r_{min} = 1 - e$. Moreover the initial condition corresponds to the pericentre. A graphical visualisation of the solution for $e = 0.05$ and $\epsilon = 1/100$ can be found in Figure 2. We solve this problem in the time in-

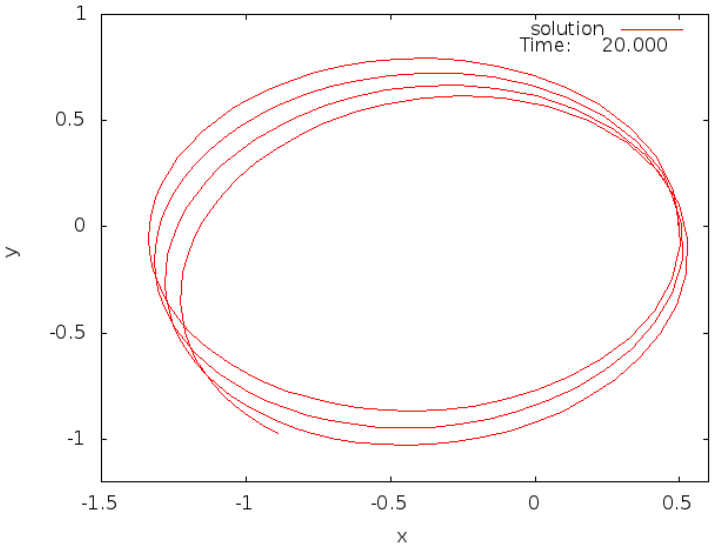
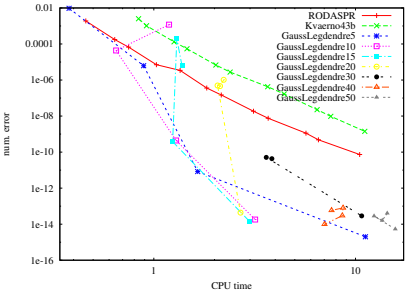


Figure 2: Visualization of the solution of perturbed Kepler's problem

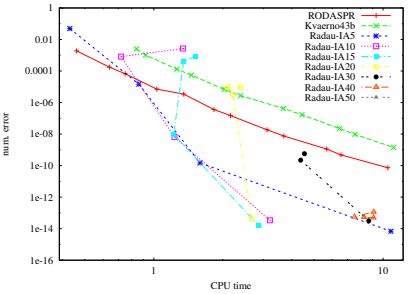
terval $(0, 1000]$ and we measure as numerical error the Hamiltonian function. The numerical results are shown in Figure 3. It can be observed that the numerical error tends to 1.0E-16 for all methods. The most efficient methods are with 10 and 15 internal stages. For methods with more than 15 internal stages the dimension of the non-linear systems is too large, and the step size restriction plays an important role. Therefore this methods are not so efficient. The RODASPR method and the Kvaerno43b method are the most inefficient since they are only of order 4 and need too many time steps.

4.2 A chemical reaction problem

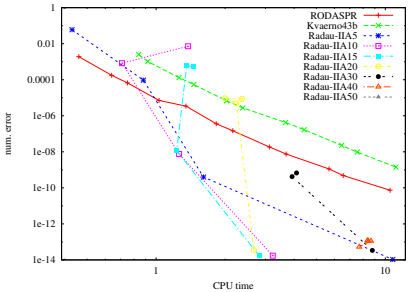
This chemical reaction problem is called E5 and can be found in the collection by Enright, Hull, and Lindberg [11] and in the book of Hairer and



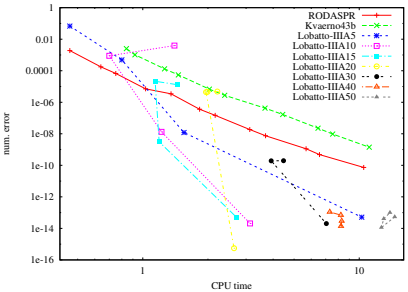
(a) Gauß-Legrendre methods



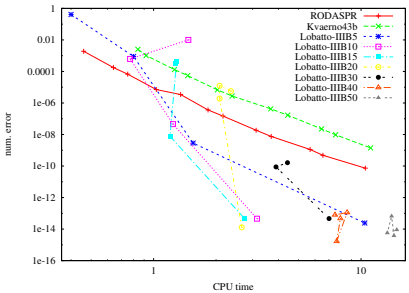
(b) Radau-IA methods



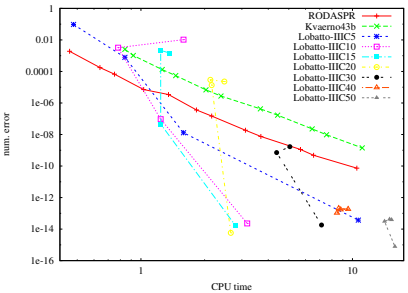
(c) Radau-IIA methods



(d) Lobatto-IIIA methods



(e) Lobatto-IIIB methods



(f) Lobatto-IIIC methods

Figure 3: Numerical results for the perturbed Kepler's problem

Wanner [14]. The equations are given by

$$\begin{aligned}
\dot{u}_1 &= -Au_1 - Bu_1u_3, \\
\dot{u}_2 &= Au_1 - MCu_2u_3, \\
\dot{u}_3 &= Au_1 - Bu_1u_3 - MCu_2u_3 + Cu_4, \\
\dot{u}_4 &= Bu_1u_3 - Cu_4
\end{aligned}$$

with the initial conditions $u_1(0) = 1,76 \cdot 10^{-3}$ and $u_i(0) = 0$, $i \in \{2, 3, 4\}$. Moreover as in [14], $A = 7,89 \cdot 10^{-10}$, $B = 1,1 \cdot 10^7$, $C = 1,13 \cdot 10^3$, and $M = 10^6$. The equations should be solved in the time interval $[0, 10^{13}]$. Note that the variables u_2, u_3, and u_4 satisfy the equation $u_2 - u_3 - u_4$. The solution is presented in Figure 4. We compute the reference solution with

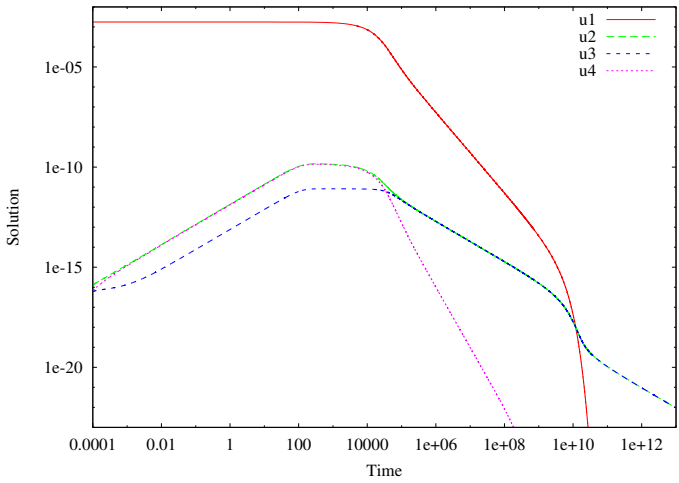
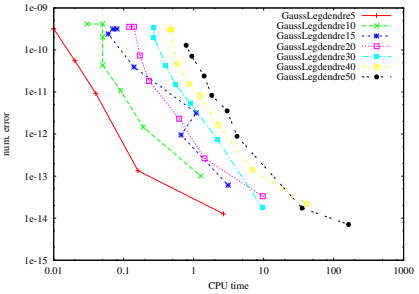
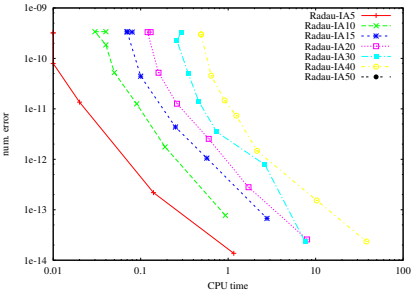


Figure 4: The solution of the problem E5

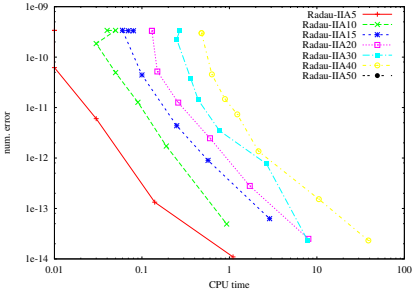
a RADAU-IIA method of order 79, i.e. with 40 internal stages, where we use a very small time stepsize. In this example we present only the fully implicit Runge–Kutta methods since the lower order methods do not reach a sufficiently high accuracy. In Figure 5 we present the numerical results. For all classes of methods it can be observed that all methods approximate the solution very well. In this case the schemes with 5 internal stages are more effective than the higher order ones.



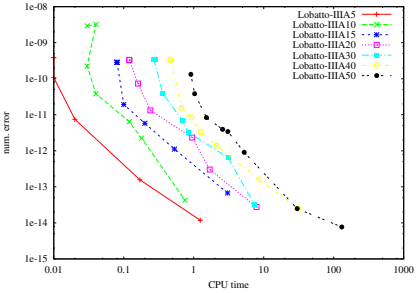
(a) Gauß-Legrendre methods



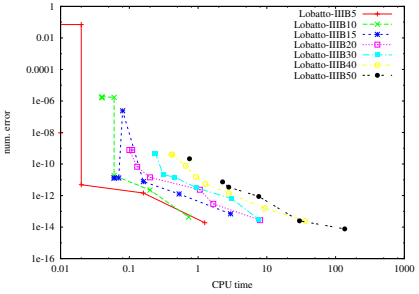
(b) Radau-IA methods



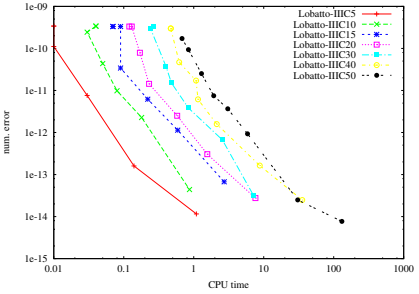
(c) Radau-IIA methods



(d) Lobatto-IIIA methods



(e) Lobatto-IIIB methods



(f) Lobatto-IIIC methods

Figure 5: Numerical results for the perturbed Kepler problem

References

[1] I.A. Abramowitz, M.; Stegun. *Handbook of Mathematical Functions.* John Wiley & Sons, 1984.

[2] T. A. Bickart. An efficient solution process for implicit Runge-Kutta methods. *SIAM J. Numer. Anal.*, 14:1022–1027, 1977. doi:10.1137/0714069.

[3] J. W. Butcher. On Runge–Kutta processes of high order. *J. Austral. Math. Soc.*, 4:179–194, 1964.

[4] J.C. Butcher. On the implementation of implicit Runge-Kutta methods. *BIT*, 16:237–240, 1976. doi:10.1007/BF01932265.

[5] J.C. Butcher. *The numerical analysis of ordinary differential equations. Runge-Kutta and general linear methods.* John Wiley & Sons, Chichester, 1987.

[6] F.H. Chipman. A-stable Runge-Kutta processes. *BIT*, 11:384–388, 1971. doi:10.1007/BF01939406.

[7] T. A. Davis. Algorithm 832: UMFPACK, an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):166–199, 2004.

[8] T. A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):165–195, 2004.

[9] P. Deuflhard. *Newton methods for nonlinear problems. Affine invariance and adaptive algorithms.* Springer-Verlag, Berlin, 2004.

[10] B. L. Ehle. High order A-stable methods for the numerical solution of systems of D.E.'s. *BIT*, 8:276–278, 1968. doi:10.1007/BF01933437.

[11] W. H. Enright, T. E. Hull, and B. Lindberg. Comparing numerical methods for stiff systems of O.D.E:s. *BIT, Nord. Tidskr. Inf.-behandl.*, 15:10–48, 1975. doi:10.1007/BF01932994.

[12] S. Gonzalez-Pinto and S. Perez-Rodriguez. Avariabletime-step-sizecode for advection-diffusion-reactionpdes. *App. Num. Math.*, 62(10):1447–1462, 2012.

[13] S. González-Pinto, S. Pérez-Rodríguez, and J.I. Montijano. Implementation of high-order implicit Runge-Kutta methods. *Comput. Math. Appl.*, 41(7-8):1009–1024, 2001. doi:10.1016/S0898-1221(00)00335-7.

[14] E. Hairer and G. Wanner. *Solving ordinary differential equations. II: Stiff and differential-algebraic problems*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1996.

[15] E. Hairer and G. Wanner. Stiff differential equations solved by Radau methods. *J. Comput. Appl. Math.*, 111(1-2):93–111, 1999. doi:10.1016/S0377-0427(99)00134-X.

[16] L. O. Jay. Inexact simplified Newton iterations for implicit Runge-Kutta methods. *SIAM J. Numer. Anal.*, 38(4):1369–1388, 2000. doi:10.1137/S0036142999360573.

[17] L. O. Jay. Solution of index 2 implicit differential-algebraic equations by Lobatto Runge-Kutta methods. *BIT*, 43(1):93–106, 2003. doi:10.1023/A:1023696822355.

[18] V. John. On the efficiency of linearization schemes and coupled multigrid methods in the simulation of a 3D flow around a cylinder. *Int. J. Numer. Methods Fluids*, 50(7):845–862, 2006. doi:10.1002/fld.1080.

[19] V. John and J. Rang. Adaptive time step control for the incompressible Navier–Stokes equations. *Comput. Methods Appl. Mech. Eng.*, 199:514–524, 2010.

[20] A. Kvaerno. Singly diagonally implicit Runge-Kutta methods with an explicit first stage. *BIT Numerical Mathematics*, 44(3):489–502, 2004.

[21] J. Lang. *Adaptive Multilevel Solution of Nonlinear Parabolic PDE Systems*, volume 16 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin, 2001.

[22] W. Liniger and R. A. Willoughby. Efficient integration methods for stiff systems of ordinary differential equations. *SIAM J. Numer. Anal.*, 7:47–66, 1970. doi:10.1137/0707001.

[23] K.-A. Mardal, T.K. Nilssen, and G.A. Staff. Order-optimal preconditioners for implicit Runge-Kutta schemes applied to parabolic PDEs. *SIAM J. Sci. Comput.*, 29(1):361–375, 2007. doi:10.1137/05064093X.

[24] J. Martín-Vaquero. A 17th-order radau IIA method for package RADAU. Applications in mechanical systems. *Comput. Math. Appl.*, 59(8):2464–2472, 2010. doi:10.1016/j.camwa.2009.12.025.

[25] T. K. Nilssen, G. A. Staff, and K.-A. Mardal. Order optimal preconditioners for fully implicit Runge-Kutta schemes applied to the bidomain equations. *Numer. Methods Partial Differ. Equations*, 27(5):1290–1312, 2011. doi:10.1002/num.20582.

[26] S. Perez-Rodriguez, S. Gonzalez-Pinto, and B.P. Sommeijer. An iterated Radau method for time-dependent PDEs. *J. Comput. Appl. Math.*, 231(1):49–66, 2009. doi:10.1016/j.cam.2009.01.020.

[27] A. Prothero and A. Robinson. On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Math. Comp.*, 28:145–162, 1974.

[28] J. Rang. An analysis of the Prothero-Robinson example for constructing new DIRK and ROW methods. *Journal of Computational and Applied Mathematics*, 0(0):–, 2013. doi:http://dx.doi.org/10.1016/j.cam.2013.09.062.

[29] J.M. Sanz-Serna and M.P. Calvo. *Numerical Hamiltonian problems.*, volume 7 of *Applied Mathematics and Mathematical Computation*. Chapman & Hall, London, 1994.

[30] K. Strehmel and R. Weiner. *Linear-implizite Runge–Kutta-Methoden und ihre Anwendung*, volume 127 of *Teubner-Texte zur Mathematik*. Teubner, Stuttgart, 1992.

[31] K. Strehmel and R. Weiner. *Numerik gewöhnlicher Differentialgleichungen*. Teubner, Stuttgart, 1995.

Technische Universität Braunschweig

Informatik-Berichte ab Nr. 2011-07

2011-07	H. Cichos, S. Oster, M. Lochau, A. Schürr	Extended Version of Model-based Coverage-Driven Test Suite Generation for Software Product Lines
2011-08	W.-B. Pöttner, J. Morgenroth, S. Schildt, L. Wolf	An Empirical Performance Comparison of DTN Bundle Protocol Implementations
2011-09	H. G. Matthies, A. Litvinenko, O. Pajonk, B. V. Rosić and E. Zander	Parametric and Uncertenty Computations with Tensor Product Representations
2011-10	B. V. Rosić, A. Kučerová, J. Sýkora, A. Litvinenko, O. Pajonk and H. G. Matthies	Parameter Identification in a Probabilistic Setting
2011-11	M. Espig, W. Hackbusch, A. Litvinenko, H. G. Matthies and E. Zander	Efficient Analysis of High Dimensional Data in Tensor Formats
2011-12	S. Oster	A Semantic Preserving Feature Model to CSP Transformation
2012-01	O. Pajonk, B. V. Rosić and H. G. Matthies	Deterministic Linear Bayesian Updating of State and Model Parameters for a Chaotic Model
2012-02	B. V. Rosić and H. G. Matthies	Stochasticc Plasticity - A Variational Inequality Formulation and Functional Approximation Approach I: The Linear Case

2012-03	J. Rang	An analysis of the Prothero–Robinson example for constructing new DIRK and ROW methods
2012-04	S. Kolatzki, M. Hagner, U. Goltz and A. Rausch	A Formal Definition for the Description of Distributed Concurrent Components - Extended Version
2012-05	M. Espig, W. Hackbusch, A. Litvinenko, H. G. Matthies and P. Wähnert	Efficient low-rank approximation of the stochastic Galerkin matrix in tensor formats
2012-06	S. Mennike	A Petri Net Semantics for the Join-Calculus
2012-07	S. Lity, R. Lachmann, M. Lochau, I. Schaefer	Delta-oriented Software Product Line Test Models - The Body Comfort System Case Study
2013-01	M. Lochau, S. Mennicke, J. Schroeter und T. Winkelmann	Extended Version of 'Automated Verification of Feature Model Configuration Processes based on Workflow Petri Nets'
2013-02	S. Lity, M. Lochau, U. Goltz	A Formal Operational Semantics of Sequential Function Tables for Model-based SPL Conformance Testing
2013-03	L. Giraldi, A. Litvinenko, D. Liu, H. G. Matthies, A. Nouy	To be or not to be intrusive? The solution of parametric and stochastic equations – the "plain vanilla" Galerkin case
2013-04	A. Litvinenko, H. G. Matthies	Inverse problems and uncertainty quantification
2013-05	J. Rang	Improved traditional Rosenbrock–Wanner methods for stiff ODEs and DAEs
2013-06	J. Koslowski	Deterministic single-state 2PDAs are Turing-complete

2014-01	B. Rosić, J. Diekmann	Stochastic Description of Aircraft Simulation Models and Numerical Approaches
2014-02	M. Krosche, W. Heinze	A Robustness Analysis of a Preliminary Design of a CESTOL Aircraft
2014-03	J. Rang	Apdative timestep control for fully implicit Runge–Kutta methods of higher order